Adaptive pixel/patch-based synthesis for texture compression

Fabien Racapé, Simon Lefort, Edouard Francois, Marie Babel, Olivier Déforges

ADAPTIVE PIXEL/PATCH-BASED SYNTHESIS FOR TEXTURE COMPRESSION

Fabien Racape, Simon Lefort, Edouard Francois

Technicolor Research and Innovation
Video Processing and Perception
Cesson-Sevigne, France

Marie Babel, Olivier Deforges

Universite Europeenne de Bretagne
INSA, IETR, UMR 6164
Rennes, France

ABSTRACT

This paper presents an adaptive scheme for synthesizing missing textured regions. In synthesis-based compression approaches, large textures are removed at encoder side and filled in at decoder side. This work proposes a synthesizer in which both complementary pixel-based and patch-based approaches are used. According to results shown by synthesis algorithms, patch-based and pixel-based approaches are efficient with different kinds of texture. Two algorithms are adapted to the region synthesis context: the sample patch is built from surrounding texture and potential anchor blocks inside the removed region; the scan order depends on a confidence map in order to take advantage of the designed patch; DCT-descriptors provide the minimum size of matching window which is a required parameter for both synthesizers. Finally an "a posteriori" gradient based assessor enables the synthesizer to switch between algorithms.

Index Terms— texture; synthesis; characterization.

1. INTRODUCTION

In State-of-the-art compression schemes, such as H.264/AVC, pixel-wise redundancies are reduced by using predictions and transform domain operations. However, spatio-temporal approaches, exploiting redundancy based on the Mean Squared Error (MSE) criterion, are not able to take visual redundancy into account. Detailed textures may seem near stationary to the Human Visual System, but totally irregular according to MSE criterion. Meanwhile, texture synthesis algorithms [1, 2] have shown great results. The purpose of new coding schemes is to detect regions where exact positions of texture patterns are irrelevant to the human eye. The whole region is not encoded since only a few patterns are sufficient to synthesize a satisfactory region. An interesting framework has been designed in [3] where some 8x8 blocks are removed at the encoder side and synthesized by the decoder. Texture synthesis is performed using the algorithm presented in [2], however using 8x8 blocks offer poor overlap for the patch-based technique. The work presented in [4] proposes a closed-loop analysis-synthesis approach. As in [3], groups of pictures (GOP) are considered for a spatio-temporal scheme. Each potential region from a GOP is both analyzed and synthesized, using side information from the texture analyzer. A first synthesizer is designed for *Rigid textures* with global motion, which has a great similarity to global motion compensation (GMC). Another synthesizer, inspired by the patch-based approach developed in [2], processes *Non-rigid textures* with local and global deformations. The scheme has recently been upgraded in [5] with photometric corrections, using Poisson editing and covariant cloning. However, Y. Liu [6] pointed out that synthesis algorithms work more efficiently with different kinds of textures and addressed the specific case of near regular textures.

This paper presents a framework using both complementary pixel-based and patch-based approaches. The scheme includes a texture characterization step, based on DCT-domain descriptors, that outputs the approximated feature size of texture patterns to be synthesized. The two following texture synthesis algorithms have inspired this work: the basic pixel-based approach developed in [1] and the algorithm presented in [2] for the patch-based synthesis.

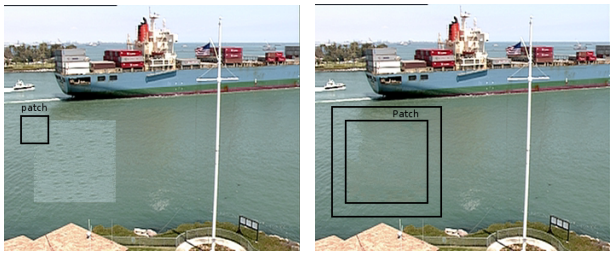
This paper is organized as follows. In section 2 is introduced the texture analysis at encoder side. In sections 3 and 4 are respectively described pixel-based and patch-based chosen algorithms. Section 5 finally presents some experimental results and the switching method between both approaches.

2. TEXTURE ANALYSIS

This section focuses on the analysis at the encoder side which aims at characterizing texture patterns and designing the removed regions and sample patches for synthesis. Note that the segmentation has been performed by the region extraction scheme presented in [7].

2.1. Texture characterization

This part of the scheme outputs the required parameters in order to optimize the synthesizers. Chosen descriptors are derived from those described in [8] which are computed from a Fourier transform. Since the characterization requires little sizes of block, i.e. inferior to 32 pixels wide, DCT domain has been preferred. Indeed, Fourier transform outputs descriptors



a) square patch b) chosen patch

Fig. 1. Texture patch design.

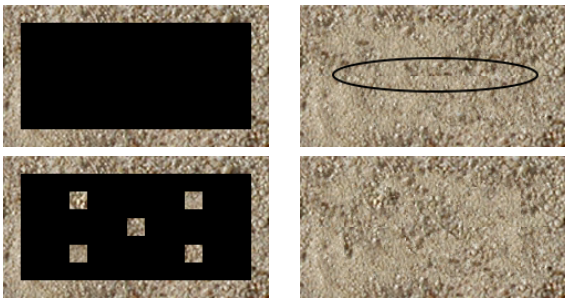


Fig. 2. Anchor blocks. Up: synthesis of an empty region. Bottom: synthesis with anchor blocks.

with a lower resolution in frequency, which is a problem to describe a signal from a small block. Like in [8], descriptors are computed from concentric circles, except that they are integrated on a quarter of circle. Their computation follows

$$D_{DCT}(\lambda) = \int_{\theta=0}^{\pi/2} |C(\lambda, \theta))|^2 \, d\theta \tag{1}$$

where C corresponds to the DCT coefficient at position $(\lambda cos\theta, \lambda sin\theta)$. The DC coefficient is not taken into account in order to be invariant to mean luminance variations, so the descriptors vector has the size of DCT block minus one coefficient. According to [1, 2] and our experiments, the size of neighborhoods has to be greater than the texture pattern, in order to produce visually good results. By experimenting a large set of arbitrary chosen texture patches, we noticed that if the first coefficient (low frequency) is greater than any others in the descriptors vector, the block size can be considered smaller than the pattern. Indeed, the first coefficient of each block corresponds to a single variation of luminance over the block used for DCT computation. The output size parameter corresponds to the minimum size NxN of DCT block that provides a non-monotone decreasing vector, which is given by

$$N_{out} = \min_{N} \left\{ B_{DCT}^{N \times N} | D(1) \neq \max_{\lambda=[2;N-1]} \{D(\lambda)\} \right\} \tag{2}$$

where B is a DCT block of size NxN. Thus, an approximate size is given by testing descriptor sizes from 4x4 to 32x32 pixels.

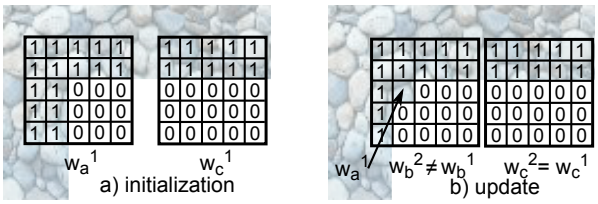


Fig. 3. Confidence map order. Confidences w_a, w_b and w_c are computed for center pixels from equation 4 using neighboring confidences. At initialization $w_a = 0.64$ for instance. After synthesizing pixel a, w_b is updated by assigning w_a^1.

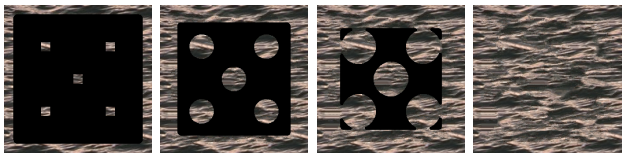


Fig. 4. Pixel-based synthesis order.

2.2. Texture patch design

Texture synthesizers from the literature process with a rectangular input patch. Its relevance is essential to ensure an efficient synthesis in this particular context. One notices that cropping a patch close to the texture region to be synthesized can lead to inconsistencies between candidate pixels and previously reconstructed borders. Figure 1 a) shows a case in which there is a variation of luminance, leading to a failing synthesis. We propose to define the patch as the surrounding MBs of the region to be synthesized. This source patch for synthesis is depicted in figure 1 b).

2.3. Encoding strategic anchor blocks

Large textured regions require anchor preserved blocks in order to prevent artefacts. After experiments, it has been decided to encode some anchor macroblocks (MB) at strategic locations depicted in figure 2 since the region size exceeds a fix number of MBs in width or height. Coupled with a confidence-based synthesis order described in next section, they prevent visible seams at synthesis junctions. The next two sections describe the synthesis tools for filling in these regions, using information provided by descriptors.

3. PIXEL BASED TEXTURE SYNTHESIS

The pixel-based synthesizer derives from the basic scheme presented in [1]. This algorithm relies on the matching between current pixels' neighborhood and those in a texture patch, using the Sum of Squares Error (SSE) as criterion. In order to improve the research process, the exact neighborhood matching described in [9] is used. The following describes the adaptation to the context of removed regions.

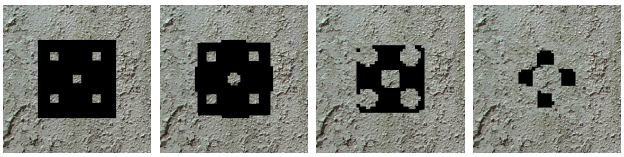


Fig. 5. Patch-based synthesis order.

Fig. 6. a: The red hatched region is used to search the best matching patch. b: the chosen patch is pasted on the new texture. c: the best cuts are found between overlapping regions

3.1. Adaptive neighborhood size using texture characterization

Texture characterization enables the synthesizer to get an approximate neighborhood. In order to refine the neighborhood sizes for matching, a set of neighborhood sizes are tested, for example 7x7 and 9x9 if the best descriptor is 8x8 large. The chosen distance minimizes the norm distance

$$D_{m,n}(p,c) = \frac{\sum_{k=0}^{N_{m,n}} (p_k - c_k)^2}{N_{m,n}} \quad (3)$$

where $N_{m,n}$ is the number of pixels contained in the neighborhood of current pixel, m and n respectively representing its height and width. c_k and p_k denote the kth pixel's luminance in the current neighborhood and the considered one in the patch respectively. Considering various neighborhood sizes enables to better catch texture patterns size and shape.

3.2. Confidence-based synthesis order

The goal is here to exploit available whether previously decoded or synthesized pixels which are, at this point, the only confident data. A confidence map has been designed to synthesize the most reliable pixel at each time. The map building law is depicted in figure 3, where initial computing (a) from previously decoded pixels is represented on the left side and the update during synthesis on the right side. The neighborhood, used for texture synthesis, serves for computing a map on which previously decoded pixels are initialized with a unique confidence value. Thus, confidence in pixels to be synthesized is computed following:

$$w(i,j) = \frac{1}{N^2} \sum_{k=-N/2}^{N/2} \sum_{l=-N/2}^{N/2} w(i+k, j+l). \quad (4)$$

The map is then updated (b) by assigning the previously computed $w(i,j)$ on its location.

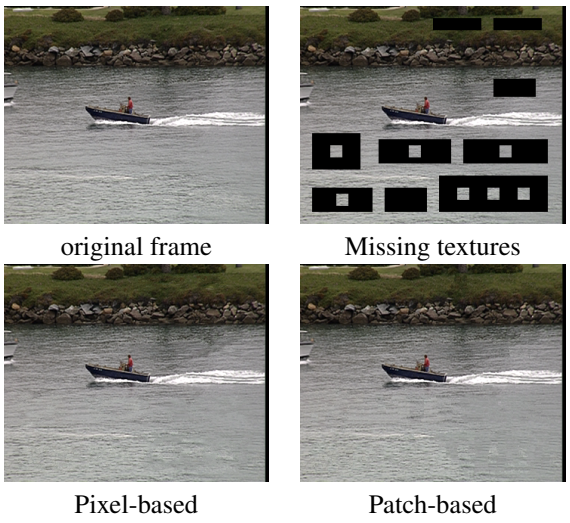


Fig. 7. Results on the *Coastguard*.

4. PATCH BASED TEXTURE SYNTHESIS

The patch based method used can be decomposed into two main steps. The synthesis order, depicted in figure 5, is based on the confidence map described in section 3.2

4.1. Finding the best matching patch

Each patch added to the output texture is chosen by comparing the already known parts, depicted in figure 6 a), with all the possible positions in the input patch. The SSE criterion is used to find the best matching area. That is, the patch size is determined by previously described DCT-descriptors.

4.2. Cutting the patch for a seam-less texture

Most of the time, the added patch does not match exactly the already synthesized texture, then a second step consists on cutting the patch with the texture using a graphcut method like described in [2]. Costs between overlapping patches is computed from the extended version in [2] using gradients. Figure 6 presents a typical case due to surrounding blocks and synthesis order. The next section provides some results and proposes a switch method to optimize the synthesis.

5. COMPARING AND SWITCHING ALGORITHMS

This section aims at comparing and presenting how to choose the best algorithm for each region.

5.1. Results and comparison

One has now to determine whether to use patch- or pixel-based technique depending on textures' characteristics. Then

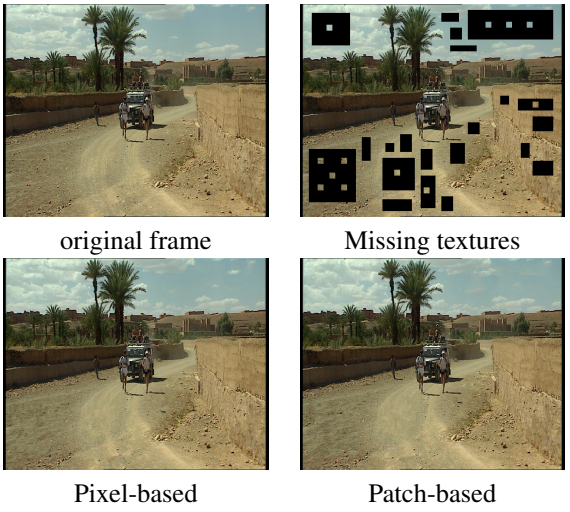


Fig. 8. Results on the *Morocco Trial* sequences.

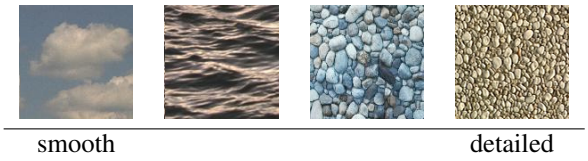


Fig. 9. Texture spectrum for switching synthesizers

an experimental classification of textures in our context is presented in figure 9. Patch-based method provides good results with detailed textures but fails with smoother ones, creating edge artefacts. Reversely, pixel-based synthesis is well-adapted to smooth variations but fails when producing large detailed patterns. According to experimental results depicted in figures 7 and 8, patch-based method seems to give good results on grass and trodden earth but fails to synthesize water or clouds where pixel-based method produces better results.

5.2. Switching between algorithms

Since the patch-based approach produces visible artefacts when failing, an a posteriori method is proposed. Three gradient-based criteria are proposed in order to quantify the amount of created artefacts. ΔGl denotes the gradient's differences at cut locations beten synthesized and original images, while ΔGh and ΔGv represent horizontal and vertical mean Sobel gradients' differencies, again between output and input textures. After experiments on a large set of textures, maximum thresholds $\{5; 5; 20\}$ have been fixed for ΔGh, ΔGv and ΔGl respectively to determine acceptable synthesis. Gl characterizes the located seams to determine if the cuts are hidden enough, while Gh and Gv aim at pointing out such oriented edges, highly detectable by the Human Visual System, like in figure 10 c). As the texture is expected synthesizable by characterization step, the pixel-based synthesizer is chosen for high gradients.

Texture	ΔGh	ΔGv	ΔGl
a	-0.3	-0.8	0.93
b	-3.5	-4.1	17.6
c	-12.3	11.1	28.7
d	-4.0	15.3	60.7

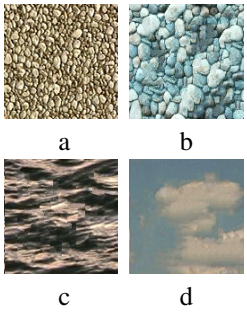


Fig. 10. Resulting gradients for the 4 textures on the right.

6. CONCLUSION AND FUTURE WORK

This paper presents an intra-frame compression framework using two kinds of texture synthesis, according to their favorite types of texture. The switch between algorithms is done a posteriori since it is based on patch-based synthesis artefacts measurement. Ongoing research focuses on the descriptor-based segmentation as well as the characterization process of detected textures in order to improve synthesizers' adaptation.

7. REFERENCES

[1] L.-Y. Wei and M. Levoy, "Fast texture synthesis using tree-structured vector quantization," in *Proceedings of ACM SIGGRAPH*, New York, USA, 2000, pp. 479–488.

[2] V. Kwatra et al., "Graphcut textures: image and video texture synthesis using graph cuts," in *Proceedings of ACM SIGGRAPH*, 2003, pp. 277–286.

[3] C. Zhu et al., "Video coding with spatio-temporal texture synthesis," in *IEEE ICME*, 2007, pp. 112–115.

[4] P. Ndjiki-Nya, T. Hinz, and T. Wiegand, "Generic and robust video coding with texture analysis and synthesis," in *2007 IEEE ICME*, 2007, pp. 1447–1450.

[5] P. Ndjiki-Nya, D. Doshkov, and M. Koppel, "Optimization of video synthesis by means of cost-guided multimodal photometric correction," *ISPA*, 2009.

[6] Y. Liu and Y. Tsin, "The promise and the perils of near-regular texture," *International Journal of Computer Vision*, vol. 62, pp. 1–2, 2002.

[7] O. Deforges, M. Babel, L. bedat, and J. Ronsin, "Color lar codec: a color image representation and compression scheme based in local resolution adjustment and self-extracting region representation," *TCSVT*, vol. 17, no. 8, pp. 974–987, 2007.

[8] F. Smach, C. Lemaître, J.P. Gauthier, J. Miteran, and M. Atri, "Generalized Fourier descriptors with applications to objects recognition in SVM context," *Journal of Mathematical Imaging and Vision*, vol. 30, no. 1, pp. 43–71, 2008.

[9] M. Sabha, P. Peers, and P. Dutre, "Texture synthesis using exact neighborhood matching," *Computer Graphics Forum*, vol. 26, no. 2, pp. 131–142, 2007.